(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平5－73143

(43)公開日　平成５年(1993)３月26日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０５Ｄ　1/02		Ｐ　7828－3Ｈ		
		Ｓ　7828－3Ｈ		
// Ｇ０５Ｂ　13/02		Ｌ　9131－3Ｈ		

審査請求　未請求　請求項の数１(全 21 頁)

(21)出願番号　特願平3－121495

(22)出願日　平成３年(1991)５月27日

(71)出願人　000002059
神鋼電機株式会社
東京都中央区日本橋３丁目12番２号

(72)発明者　江川　隆己
三重県伊勢市竹ケ鼻100番地　神鋼電機株式会社伊勢製作所内

(72)発明者　高松　千明
三重県伊勢市竹ケ鼻100番地　神鋼電機株式会社伊勢製作所内

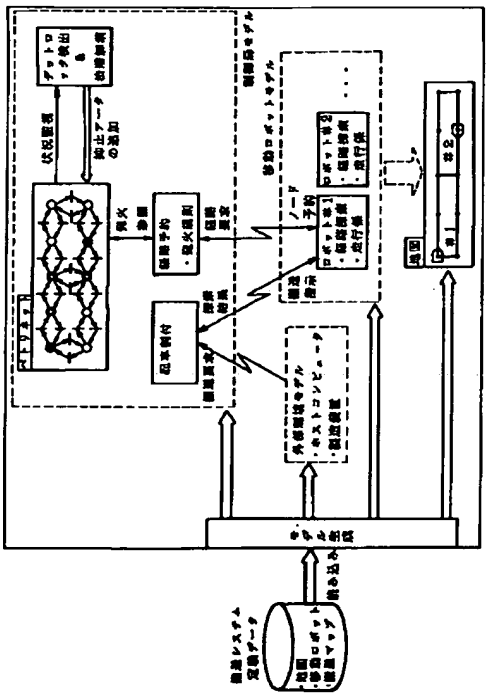
(74)代理人　弁理士　志賀　正武　(外２名)

(54)【発明の名称】　移動ロボットシステム

(57)【要約】

【目的】　衝突の発生防止を早期に行い、渋滞やデッドロックを防止する。

【構成】　衝突あるいは衝突が起り易い状態になったのを検知し、その原因となった各移動ロボットの配置状態および進行状態を検出する原因検出手段と、前記原因検出手段によって検出された原因を抑止情報として記憶する抑止情報記憶手段と、前記抑止情報に基づいて前記移動ロボットの移動を制限し、前記移動ロボットの移動制御を行う制御手段とを設けた。



1

【特許請求の範囲】

【請求項１】　衝突あるいは衝突が起り易い状態になったのを検知し、その原因となった各移動ロボットの配置状態および進行状態を検出する原因検出手段と、

前記原因検出手段によって検出された原因を抑止情報として記憶する抑止情報記憶手段と、

前記抑止情報に基づいて前記移動ロボットの移動を制限し、前記移動ロボットの移動制御を行う制御手段とを具備することを特徴とする移動ロボットシステム。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】この発明は、複数の移動ロボットと、これらの移動ロボットを制御する制御局とから構成される移動ロボットシステムに関する。

【０００２】

【従来の技術】近年、ＦＡ（ファクトリ・オートメーション）の発達に伴い、この種のシステムが各種開発され、実用化されている。この移動ロボットシステムにおいて、制御局は各移動ロボットへ無線または有線によって行先およびその行先において行う作業を指示する。制御局から指示を受けた移動ロボットは、指示された場所へ自動走行して到達し、その場所で指示された作業を行い、作業が終了した時はその場で次の指示を待つ。

【０００３】さて、この種のシステムにおいては、自動走行するロボット同士の衝突をいかに防ぐかが大きな問題である。この問題を解決する１つの方法として、１台のロボットが走行している時に他のロボットを停止させ＊

2

＊ることが考えられる。しかし、この方法の場合、衝突の可能性は非常に少なくなるが、ロボットの走行効率が極端に悪くなってしまい、複数のロボットを走行させる意味がない。そこで、各ロボットに走行の予約を行わせ、各ロボット同士の衝突を回避するようにした移動ロボットシステムが提案されるに致った。以下、この移動ロボットシステムの概略を説明する。このシステムにおいて、各移動ロボットが走行する走行路の各拠点にはノードが設定されている。各ロボットはメモリを内蔵しており、このメモリには、以下に示すように走行路について記述した地図データが記憶されている。

①ND-ENTRY: ノードの定義

②NETWORK:ノードに関するデータ。ノードの座標，タイプ（ｅｘ．スピンターン可能，移載），他のノードとの接続関係などを記述する。

③SCENE:ノード間のデータ。両端のノード，距離，走行方向（一方通行か両方向可能か），走行姿勢（前後進，横行）といったデータからなる。

④COLLISION:ロボット同志の衝突防止用のテーブル。あるノードにロボットがいるとき、他のロボットが他のあるノードにきたら衝突してしまうようなノード（干渉ノード）を、ロボットの形状とノードの座標から予め求めておき、このテーブルにセットしておく。

ロボットは、出発ノードと目標ノードが与えられると、下記（１）式によって示される評価関数により、隣接する各ノードの評価を行い、該評価結果に従って、次に進むべきノードを順次決定していく。

$$H = Wa\ A(Vj) + Wc\ C(Vi, Vj) \qquad (1)$$

ここで、Vi が探索中において現在着目しているノードで、Vj は次の候補ノードである。また、WaおよびWcは各々評価係数であり、Wa ＝１，Wc ＝０．５となっている。また、Ａ（Vj）は、Vj と目標ノードとの正規化距離であり、Ｃ（Vi，Vj）はノードVi とノードVj との間の正規化距離である。以上のようにして、経路の探索を行った後、ロボットは、一定距離Xm を越えた最初の分岐ノードまでの走行を制御局Ｃに予約要求し、この要求が許可されることを以て、許可されたノードまで走行するという動作を繰り返し、目標ノードまで進む。ここでXm は、ロボットの速度やサイズから一意的に決まるパラメータである。一方、制御局は、予約要求された分岐ノードまでの経路に属する各ノードについて、それらが他のロボットによって予約（干渉ノードも含む）されているか否かを判断し、すべてのノードについて予約がされていなければ、予約許可を与える。他のロボットが予約していれば、その手前までしか許可しない。ロボットは、要求したノードをすべて予約できなかった場合は、予約できた最後のノードまで行って、そこで一旦停止し、他のロボットが通りすぎるのを待ったり、迂回路を捜すといった処理を行なう。

【０００４】

※【発明が解決しようとする課題】ところで、上述した従来の移動ロボットシステムは、各時点において、その直後に起り得る衝突を予測し、それを回避することが可能であったが、衝突が起り易い状態、例えば多くのロボットが狭い領域に密集した渋滞状態になるのを防ぐことができなかった。このため、特にロボットの数が多くなると、頻繁に走行路が渋滞状態となり、各ロボットが衝突回避のための動作を行う回数が多くなり、システムの効率が悪化するという問題があった。

【０００５】この発明は上述した事情に鑑みてなされたものであり、走行路が衝突の発生し易い状態になるのが未然に防止され、効率的の改善された移動ロボットシステムを提供することにある。

【０００６】

【課題を解決するための手段】この発明は、衝突あるいは衝突が起り易い状態になったのを検知し、その原因となった各移動ロボットの配置状態および進行状態を検出する原因検出手段と、前記原因検出手段によって検出された原因を抑止情報として記憶する抑止情報記憶手段と、前記抑止情報に基づいて前記移動ロボットの移動を制限し、前記移動ロボットの移動制御を行う制御手段とを具備することを特徴とする。

【0007】

【作用】上記構成によれば、衝突あるいは衝突が起り易い状態になると、その原因が検出され、抑止情報として記憶される。そして、以後、同じ原因によって衝突あるいは衝突が起り易い状態にならないように移動ロボットの制御が行われる。

【0008】

【実施例】以下、図面を参照し、この発明の実施例を説明する。

【0009】＜第１実施例＞

１．移動ロボットシステムの概要

図１はこの発明の第１実施例による移動ロボットシステムの構成を示すブロック図である。第１図において、点線で囲んだ部分、すなわち、制御局Ｃ、移動ロボットＲ、走行路Ｌが、本実施例による移動ロボットシステムに係る部分である。制御局Ｃに、ホストコンピュータＨなどから搬送要求（以下、ジョブという）が入力される。制御局Ｃは、これまでに未配車のジョブのうち適当なジョブを選択して、待機中のロボットＲ、Ｒ、…から最適なロボットを選んで、ジョブを割り付ける（配車割付）。各ロボットは、内蔵のメモリに走行路Ｈを記述した地図データを記憶している。配車割付されたロボットＲは、走行路Ｌを規定する地図データをもとに、目的地までの経路を決め、それに従って他のロボットと衝突しないように走行する。以下、本システムの各部の構成および動作を順に説明する。

２．走行路

ロボットが停止できるノード（白丸）を基本に、走行可能なノード間をつなげて（線）、走行路Ｌが構成される。走行路の形態は大きく分け、図２（ａ）に示すような梯子型走行路と図２（ｂ）に示すようなループ型走行路に分類される。以下にこれらの各走行路の特徴を列挙する。

［梯子型走行路］　走行路の両側に装置が配置され、その間が非常に狭い場合に、梯子型が良く採用される。２台がスレ違うだけの幅しかないため、走行路は必然的に、両方向になる。したがって、右からくるロボットと左からくるロボットが向かい合うといった、ロボット間の干渉が頻繁に起こり、搬送効率からみればループ型に劣る。

［ループ型走行路］　一方、搬送スペースが比較的広い場合は、一方通行のメインループを基本としたループ型になる。ステーションで移載作業（以下、作業という）中に、メインループ上の走行路を塞がないように、メインループから枝が出ていて、その先端の葉の位置で作業をする。したがって、枝の部分は両方向である。

３．ロボット

配車割付の行われたロボットは、現在位置するノードから制御局Ｃによって指示された目標ノードに致る経路を探索する。上述した従来の移動ロボットシステムと異な

り、各ロボットは予め走行路をペトリネットを用いてネットワーク表現したモデルを地図データとして記憶している。ノード間を結ぶアークには、ノード間の移動コストに相当する数値がセットされる。基本的には、このネットワーク上で最短経路アルゴリズムを用いて、出発ノードから他の全てのノードへの最短（コスト最小）経路を探索する。この経路探索は、以下の３種類の方法が状況に応じて選択された用いられる。

①最適経路探索：制御局から行き先を指示されて、動き出す前に行なう経路探索であり、他のロボットの位置や状態は考えない。上述した経路探索に相当するものである。

②準最適経路探索：走行中、他のロボットと干渉して、自分が経路変更する場合の経路探索で、邪魔なロボットのいるノード（およびその干渉ノード；以下、同様）と自分を邪魔としているロボットのいるノードをネットワークから除いて、探索する。コスト的には、最初予定していた経路より劣る経路が得られる。

③待避経路探索：自分が作業を持っていなくて、あるノードで待機中に、他のロボットの邪魔となり、待避する経路を捜す場合である。相手ロボットの経路の邪魔とならないようなノードを選択するという条件が加わる。準最適経路探索において、干渉するロボットのノードを除くため、経路が求まらないことが良くある。現行の経路探索アルゴリズムが、とにかく経路を見つけていたのに比べ、デッドロックに陥り易い。これは、「渋滞箇所でじたばたするより、一旦は諦めて、次回からはこのような状態にならないようにしよう」とするために、渋滞箇所をデッドロックにより見つけたいという発想からきている。

４．制御局

移動ロボットシステムの抱える現在の大きな問題点は、複数のロボットが特定の領域に密集し、互いに邪魔しあって動きづらくなること（渋滞）および動けなくなること（デッドロック）である。クリーンルームでは面積あたりのコストが高いため、搬送スペースを広くとれず梯子型走行路になる場合が多い。このため、従来は、一本の経路上で互いに向き合う状況が頻繁に生じ、その都度適当なロボットが制御局Ｃと通信しながら経路変更を行なっていた。このような衝突を回避する動作は時間を要し、その最中に新たなロボットがやってきて、さらに問題を複雑にすることがあり、ひどい渋滞を招いてしまう。最悪の場合、デッドロックを引き起こしていた。従って、なるべく渋滞を少なくすることが望ましい。この移動ロボットシステムは、制御局が行う経路予約処理において、制御局が周囲の状況を見て、渋滞の発生が予測される場合、経路予約を拒否するようにしたものである。以下、その具体的な説明を行う。最初に経路予約の難しさについて言及し、経路予約条件の自動獲得機構の必要性を指摘する。次に、自動獲得機構実現のための主

要な構成要素について説明していく。

（1）経路予約の難しさ

上述したように、ロボットRは制御局Cに経路上のノードを要求し、許可されたノードについては、進入可能とみて突き進む。ここで、制御局Cが、周囲の混雑度を考慮しないで、他のロボットがノードを予約しているか否かだけの判断に基づいて許可を出すと、渋滞、あるいは最悪の場合デッドロックに致る。この問題を解決するためには、制御局Cが周囲の状況を判断し、要求された走行を許可することによって渋滞しそうな場合は予約の許可を出さないようにする必要がある。ところが、周囲の状況というのは、以下の要因が複雑に絡み合ったものである。

❶走行路の形状－周りが十字路，三叉路，行き止まりといった場合で、見るべき範囲が違うと考えられる。

❷ロボットの台数－3台ではほとんど渋滞しない場合でも、4台になると渋滞するというように、台数が増えると混雑度は著しく増えていく。

❸搬送要求－特定のノードでの作業指示が集中すると、必然的にひどい渋滞を招く。

さらに移動ロボットシステムは、千差万別の顧客ニーズに対応しなければならない。したがって、人間が予め渋滞状況を予測し、渋滞を引き起こさないような経路予約を、与えるのは不可能である。このため、個々の移動ロボットシステムの特性に応じて、経路予約処理の適切な判断条件を自動的に獲得する機構が必要である。

（2）経路予約状況のペトリネットによるモデル化

経路予約処理のための判断条件を得るための自動獲得機能を実現するため、本実施例では、経路予約状況のペトリネットによるモデル化を行っている。「ノードが予約されている」という状態を、プレースに対応付ける。走行路の方向性を考慮して、ノード間をトランジションで結べば、トランジションの入出力プレースの数は常に1になる。したがって、経路予約状況はペトリネットのサブクラスである状態マシン（State Machine: SM）によって自然に表現できる。また、上述した干渉ノードを表現するために、抑止アークを取り入れる。干渉ノードは、ノードの座標およびロボットの形状といった幾何的要因によって決まるので、幾何抑止アークと名付ける。ここで、簡単な例で、ペトリネットを用いてどのようにモデル化されるか説明する。

例1：図3の走行路において、ノード1，2，3，4からなるループは一方向（時計回り）でノード3と5の間は両方向とする。これに対応するペトリネットは、図4のようになる。図4におけるプレースｐｉ（ｉ＝1～5）は、図5におけるノードにｉに対応する。ノード3と5の間が両方向のため、トランジションｔ5とｔ6が加わる。ロボットがノード1にいて、ノード5まで（経路：1→2→3→5）を要求した場合、制御局は発火すべき系列｛ｔ1，ｔ2，ｔ6｝が順次発火可能か調べることになる。

例2：図5の走行路において、ノード2と3は近すぎて、ロボットが2台同時に入れないとする（ノード2と3は、干渉ノード）。また、走行路はすべて両方向とする。このとき、対応するペトリネットは、図6のようになる。ここで、図6におけるプレースＰｉは図5におけるノードｉに対応している。また、図6におけるトランジションｔｉｊは図5におけるノードｉからノードｊへの遷移を意味する。トランジションｔ12からプレースｐ3、ｔ43とｔ63からｐ2へ幾何抑止アークＩＮＨが出ている。プレースはノードと一対一対応するので、以下ではプレースをノードと同一視し、ノードと呼ぶこともある。

（3）経路予約処理

制御局Cは、ロボットRから経路要求がくると、上述の例1のように、経路要求を発火すべきトランジションの系列に変換する。この系列は、すでに発火した系列（既発火系列）とまだ発火していない系列（未発火系列）に分けられる。既発火系列は既に予約したノード列に、未発火系列はこれから予約したいノード列に関連する。ペトリネットのプレースには、プレースに相当するノードにロボットがいれば黒丸（黒トークン）、ロボットによって予約されていれば白丸（白トークン）が置かれる。したがってペトリネットのマーキングは、ノードの予約状況を表わしている。制御局Cは、このペトリネット上で、未発火系列のトランジションを順次取り出し、発火可能条件を満たせば、発火させる。発火できなければ、そこで予約処理は終る。発火によって、新たに白丸が現われたノードを、予約ノードとしてロボットに返す。実際は、分岐ノードまでを一括して予約するのが、説明の便宜上省略する。

ａ．発火可能条件：トランジションは以下の条件を全て満たすとき、発火可能であるという。

❶入力プレースに（黒または白）トークンがある。

❷出力プレースにトークンがない。

❸幾何条件：全ての幾何抑止アーク先にトークンがない。

❹経路条件：全ての経路抑止条件を満たさない。（これについては、後述する。）

ｂ．発火処理：発火可能なトランジションは、瞬時発火して、出力プレースに白トークンが現われる。

実システムにおけるロボットの移動によって、ペトリネット上のマーキングも当然変わるが、経路予約処理に関係ないので、説明を省く。

（4）渋滞回避処理

以上の方式にしたがって制御し、デッドロックが発生したら、その要因を推測し、今後のデッドロックが生じないような経路抑止アークを、ペトリネット上に追加していく。説明にはいる前に、言葉の定義をしておく。

Ｒ：ロボットの集合

7

集合Rに含まれる各ロボットriは、以下のデータを持つ。
Place（ri）：riのいるノード
Goal（ri）：riの目的ノード
Path（ri）：目的ノードへ至る途中の経路上のノード列
Obstacle（ri）：riが邪魔とするロボット
一方、制御局Cは、各ロボットについて、以下のデータを記憶している。
LFT（ri）：riの最後に発火したトランジション＊10
（1）ri+1＝Obstacle（ri）　　（1≦i＜n）
（2）r1＝Obstacle（rn）
（1），（2）を満たす｛r1，r2，・・・，rn｝を、核の競合ロボット集合（NCS）と呼ぶ。n＝2のときはお見合いで、n＝3のときは三竦みに該当する。
b．渋滞回避アルゴリズム
デッドロックを検出したら、以下の手順により渋滞回避のための経路抑止アークをつける。
（A）競合ロボット集合（CS）の決定
①核の競合ロボット集合（NCS）を求める。
②CSを、以下の手順により求める。
I．CS←NCSとおく。
II．Rに含まれるrのうち次の条件を満たすrを求める。rがCSに含まれておらず、かつObstacle（r）がCSに含まれる。
III．もし、rがなければ、終了。さもなければ、rをCSに加えて、IIへ戻る。
（B）渋滞原因の推定
①CSに含まれる各riについて、TFT（ri）を相互に比較し、最後に発火したトランジションLFT（r）を見つける。
②rにとって邪魔なロボットの集合CS（r）を、以下の手順により求める。
I．Old- Path（r）から、順次ノードpを取り出す。
II．CSに含まれるrjのうち次の条件を満たすrjがあればIII へ進み、さもなければIへ戻る。
p＝Place（rj）
III．CS（r）＝｛rj｝とおく。
IV．CSに含まれるrkのうち次の条件を満たすrkがあればVへ進み、さもなければ終了。
1）rk≠r
2）rkがCS（r）に含まれない
3）rk＝Obstacle（ri）or　ri＝Obstacle（rk），riはCSに含まれる
V．CS（r）にrkを追加して、IVへ戻る。
（C）経路抑止アークの追加
トランジションLFT（r）に、次の経路抑止アークをつける。
主経路抑止アーク：Place（rj）へ、実線の抑止アー※クをつける。
副経路抑止アーク：全てのPlace（rk）へ、点線の抑止アークをつける。
このようにして得られる主経路抑止アークおよび副経路抑止アークをまとめて、一つの経路抑止条件（図7参照）とする。制御局が経路予約処理を行なうとき、そのロボットの経路は知っている。経路抑止アークの意味するところは、制御局があるノードの予約処理を行なうとき、すなわち対応するトランジションが発火可能かどうか決定する際に、主経路抑止アーク先のノードを通るときには、そのノードが予約されていてかつ副経路抑止アーク先の全てのノードも予約されているならば、渋滞を引き起こすため、発火を抑止する。点である。したがって、上述の発火可能条件における経路抑止条件は以下のようになる。
［経路抑止条件］
ロボットが経路抑止アークの主経路抑止アーク先のノードを通る
かつ主経路抑止アーク先のノードが予約されている
かつ副経路抑止アーク先の全てのノードも予約されている

5．　制御局の詳細な構成
制御局Cは、走行路Lを記述したペトリネットモデルを記憶しており、このペトリネットモデルを用い、以上述べたようにして渋滞の原因となるものを検出し、渋滞の再発を防止するための経路抑止条件（経路抑止アーク）を発生し、ペトリネットモデルに追加する機能を有している。そして、システムの稼働時は、ロボットRから予約要求があった場合に制御局Cによってペトリネットモデルを用いたシミュレーションが行われる。そして、このシミュレーションの結果に従って移動ロボットからの走行要求に対する許可の制限が行われ、渋滞の未然防止が行われる。経路抑止条件の発生およびペトリネットモデルへの追加を行う方法としては、次の2通りが可能である。
方法1：移動ロボットシステムの稼働前に、種々の条件でシミュレーションを行って渋滞を検出し、その渋滞を防止するための経路抑止条件をペトリネットモデルに追

8

＊TFT（ri）：LFT（ri）が発火した時刻
Old- Path（ri）：LFT（ri）が発火した時点で、riが持っていた目的ノードまでのノード列
a．デッドロック検出方法
以下の二つの場合をデッドロックとして検出する。
デッドロックDL1：準最適経路探索したが、経路が見つからない。
デッドロックDL2：n台のロボット｛r1，r2，・・・，rn｝が、以下の状態で止まっている。ただし、n≧2。

加する。そして、実際の稼働時は、経路抑止条件の追加されたペトリネットモデルを用いて、ロボットからの走行要求を許可した場合のシステムの状態の遷移をシミュレーションし、該シミュレーション結果に従って走行要求に対する許可を制限する。

方法２：移動ロボットシステムの稼働中に各移動ロボットを監視する。そして、渋滞が検知された時点で、その渋滞を防止するための経路抑止条件を求め、蓄積する。そして、以後は蓄積された経路抑止条件に従い、ロボットからの走行要求に対する許可を制限する。

方法１による渋滞防止を行う場合について説明する。渋滞は、Symbolics(Common Lisp)上に作成されたシミュレータによってシミュレーションされる。このシミュレータの構成を、図８に示す。主な振舞いは、以下の通りである。

①地図（走行路の形状）・ロボットの台数・搬送マップ（積み－卸しステーションとその要求頻度）といった、搬送システムを定義するデータを入力として、外部環境・制御局・移動ロボットモデルを作る。

②ペトリネットモデルによって、経路予約状況とノード予約の判断条件を、構造的かつ視覚的に表現する。

③搬送マップにしたがって、搬送要求を生成し、制御局モデルの配車割付処理へ送る。制御局は、搬送要求の積みステーションを目標ノードとして、各ロボットへ経路探索指示を出す。ロボットは、現在地から目標ノードまでの経路を探し、その評価値を制御局に返す。制御局は、評価値の最も良いロボットに配車を割り付ける。

④配車割付されたロボットは、予定していた経路にしたがって、制御局に経路要求を出しながら走行する。これまで説明したように、途中で他のロボットと干渉した場合、経路を変えることもある。提案方式では、経路探索を含めた複数台制御アルゴリズム（渋滞解消機能）を、実機よりも簡単にし、ちょっと複雑な渋滞に対して解消不能（デッドロック）になるようにする。

⑤シミュレーション中は常に、ロボットの状態を監視し、デッドロックが生じたらその時点でシミュレーションを中断する。そして、上述した渋滞回避方法によって、そのデッドロックを次回からは防ぐような経路抑止アークを、ペトリネットモデルに加える。

⑥再び、最初からシミュレーションを行い、③～⑤の処理を何回も繰り返す。

以上により、最終的には渋滞をほとんど起こさないような経路予約処理の構造が、ペトリネットモデル上に構築される。

［シミュレーション例］梯子型とループ型に対して、図９と図１１のような走行路を例に、シミュレーション実験を行なった。搬送要求はそれぞれ図１０と図１２である。シミュレーション中にデッドロックを検出したら、上述の渋滞回避方法に従って、経路抑止アークをペトリネット上に付加した後、再度最初からシミュレーション

を行なう。全ての搬送要求を処理できるようになるまで、これを何回も繰り返す。図９の場合について、経路抑止アークの獲得例を説明する。

「例１」（図１３および図１４参照）

◇抑止アークの状態

経路抑止アークはどこにもついてない（初期状態）。

◇デッドロックの状況

ロボット０はロボット３を邪魔とし、ロボット３はロボット１を邪魔とし、ロボット１はロボット２を邪魔とし、ロボット２はロボット０を邪魔とし、４台すくみのデッドロックを検出する。

◇抑止アークの追加

現ノードを最後に予約したロボット探す。この場合、ロボット０が最後にノード３を予約したとすると、トランジション５が最後に発火したので、トランジション５から、ロボット０が邪魔としているロボット３の現ノード５へ主経路抑止アーク、ロボット３が邪魔としているロボット１の現ノード１５とロボット１が邪魔としているロボット２の現ノード１２へ副経路抑止アークがつく。これにより、トランジション５を発火したいとき、ノード５とノード１５とノード１２が予約されていて、ノード５の方へ進みたいときはトランジション５を発火できなくなる。同じ配車要求で再びシミュレーションを行なうとロボット０はノード２で止まり、デッドロックは生じない。

「例２」（図１５および図１６参照）

◇抑止アークの状態

トランジション１４からノード１２へ主経路抑止アーク、ノード３へ副経路抑止アークがついている。（初期状態）

◇デッドロックの状況

ロボット０はトランジション１４の抑止アークのためにロボット２を邪魔とし、ロボット２はロボット３を邪魔とし、ロボット３はロボット０を邪魔とし、３台すくみのデッドロックを検出する。

◇抑止アークの追加

現ノードを最後に予約したロボットを探す。この場合、ロボット２が最後にノード１２を予約したとすると、トランジション３２が最後に発火したので、トランジション３２からロボット２が邪魔としているロボット３の現ノード３へ主経路抑止アーク、ロボット３が邪魔としているロボット０の現ノード５へ副経路抑止アークがつく。これにより、トランジション３２を発火したいとき、ノード３とノード５が予約されていて、ノード３の方へ進みたいとき、トランジション３２を発火することはできなくなる。同じ配車要求で再びシミュレーションを行なうと、ロボット２はトランジション３２についた抑止アークのためにノード１５で止まるが、さらにデッドロックがおこる。その結果は次の通りである。

「例２の続き」（図１７および図１８）

◇抑止アークの状態
トランジション14からノード12へ主経路抑止アーク、ノード3へ副経路抑止アークがついている。トランジション32からノード3へ主経路抑止アーク、ノード5へ副経路抑止アークがついている。（初期状態）
◇デッドロックの状況
ロボット0はロボット2を邪魔とし、ロボット2はトランジション32の抑止アークのためにロボット3を邪魔とし、ロボット3はロボット0を邪魔とし、3台すくみのデッドロックを検出する。
◇抑止アークの追加
現ノードを最後に予約したロボットを探す。この場合、ロボット2が最後にノード15を予約したとすると、トランジション39が最後に発火したので、トランジション39から、ロボット2が邪魔としているロボット3の現ノード3へ主経路抑止アーク、ロボット3が邪魔としているロボット0の現ノード5へ副経路抑止アークがつく。これにより、トランジション39を発火したいとき、ノード3とノード5が予約されていて、ノード3の方へ進みたいとき、トランジション39を発火することはできない。同じ配車要求で再びシミュレーションを行なうと、ロボット2はノード16で止まり、デッドロックは生じない。いまの場合のデッドロックは前ページと2回の学習によって解決できることになる。
「例3」（図19および図20）
◇抑止アークの状態
経路抑止アークはどこにもついていない。（初期状態）
◇デッドロックの状況
ロボット0はノード11で作業を終えて、ノード6へ行きたい。しかし、両側をはさまれて準最適経路が見つからない。ロボット1は目的ノードが11なので準最適経路が見つからない。ロボット2も目的ノードが11なので準最適経路が見つからない。
◇抑止アークの追加
現ノードを最後に予約したロボットを探す。この場合、ロボット2が最後にノード10を予約したとすると、トランジション6が最後に発火したので、トランジション6から、ロボット2が邪魔としているロボット0の現ノード11へ主経路抑止アーク、ロボット0が邪魔としているロボット1の現ノード12へ副経路抑止アークがつく。これにより、トランジション6を発火したいとき、ノード1とノード12が予約されていて、ノード11の方へ進みたいときはトランジション6を発火できなくなる。同じ配車要求で再びシミュレーションを行なうとロボット2はノード2で止まり、デッドロックは生じない。次に図9の走行路と図10の搬送要求に従って、シミュレーションを行なった結果を述べる。ロボットの台数が変わると経路抑止アークのつく結果も変わると思われたのでロボットの台数を3台と4台で実験を行なった。ロボットを3台走らせた結果は図21および図22

である。経路抑止アークは全部で6個ついている。4台すくみが起こることはないので、副抑止アークが2つついていることはない。ロボットを4台走らせた結果は図23および図24である。経路抑止アークは全部で9個付いている。トランジション9と32には3台走らせたときと同じ抑止アークがついている。トランジション11と16と41には3台のときも4台のときも抑止アークがついているが、抑止アークがついているノードが異なっている。また、4台すくみがおこっているので副経路抑止アークが2つついているトランジションがある。走行路、搬送要求が同じでも、ロボットの台数が変わると状況が変わってくるので、違った結果になったと思われる。次に図11の走行路と図12の搬送要求に従って、シミュレーション結果を行なった結果を述べる。ロボットの台数が変わると経路抑止アークのつく結果も変わると思われたのでロボットの台数を3台と4台と5台で実験を行なった。ロボットを3台走らせた結果は図25および図26である。経路抑止アークは全部で13個ついている。副経路抑止アークがないところは袋小路ノードすなわち作業ノードの入口での競合である。トランジション5，7，9には同じ抑止アークがついている。これはノード7と21で競合がおこっているところにノード6に第三者のロボットがきて3台の競合となり、第三者の経路上に3つの抑止アークがついた結果である。ロボットを4台走らせた結果が図27および図28である。経路抑止アークは全部で7個ついている。どの抑止アークも副経路抑止アークがない。つまり、袋小路ノードの入口でしか競合がおこっていない。ロボットを5台走らせた結果は図29および図30である。経路抑止アークは全部で11個ついている。副経路抑止アークがあるトランジションもあり、袋小路ノード以外でも競合がおこなっていることがわかる。梯子型に比べて競合が少ないので、違いがよくわかりにくいが、ループ型でも同じ走行路、搬送要求においてロボットの台数が変わると経路抑止アークの結果は変わるといえると思われる。
［評価］シミュレーション実験を通して得られた知見を述べる。シミュレーション実験した結果、渋滞解消アルゴリズムは、基本的に当初想定していたとおりの振舞いを示し、最初は渋滞を引き起こしていたのが、“学習”を進めるにしたがって次第にスムーズに走るようになり、最後には渋滞を引き起こさず全ての搬送要求を処理することが可能となった。梯子型，ループ型の走行路共、規模が小さく、この上で4～5台のロボットを走らせるという条件設定は、狭い領域にロボットが密集し易いため、かなり厳しい。このような条件の下で、渋滞を招かないような制御構造をペトリネット上に構築できたことは、提案した方式の有効性示す一つの証左と考えられる。このような認識にたって、経路予約方式を提案し、シミュレーション実験した結果、渋滞解消アルゴリズムは、基本的に当初想定していたとおりの振舞いを示

し、最初は渋滞を引き起こしていたのが、“学習”を進めるにしたがって次第にスムーズに走るようになり、最後には渋滞を引き起こさず全ての搬送要求を処理することが可能となった。したがって、提案方式は有効であると結論づけられる。

【００１０】<第２実施例>本実施例は、移動ロボットシステムにおいて、搬送路を記述した地図データを、ペトリネットＰＮ＝（Ｐ，Ｔ，Ｉ，Ｏ，ＧＩ，ＰＩ，Ｍ）でモデル化するものである。走行路を予約処理ロジックと関連付けて構造的に表現するため、上記第１実施例と同様なペトリネットを応用したモデルを用いる。本モデルは、プレース、トランジション、アークからなる。これらの定義および用法は上記第１実施例に準ずる。また、プレースからトランジションへ向かうアークを入力アーク、トランジションからプレースへ向かうアークを出力アークと呼ぶ。また、入力アークのプレースを入力プレース、出力アークのそれを出力プレースと呼ぶ。このモデルでは、トランジションの入力アーク，出力アークは共に１としている。また、上記第１実施例と同様、幾何抑止アーク、経路抑止アーク、トークンを用いる。黒トークン、白トークンの定義も上記第１実施例と同じである。したがって、ペトリネットＰＮは

ＰＮ＝（Ｐ，Ｔ，Ｉ，Ｏ，ＧＩ，ＰＩ，Ｍ）

ここで、Ｐ＝｛ｐ１，ｐ２，・・・ｐｍ｝　；プレースの有限集合

Ｔ＝｛ｔ１，ｔ２，・・・ｔｎ｝　；トランジションの有限集合

ＩはＰ×Ｔの部分集合であって入力アークの集合

ＯはＴ×Ｐの部分集合であって出力アークの集合

ＧＩ　；幾何抑止アークの集合

ＰＩ　；経路抑止アークの集合

Ｍ：Ｐ→｛０，１，２｝　；マーキング関数

である。Ｍは、各プレースを、０（＝トークンなし），１（＝白トークンあり），２（＝黒トークンあり）といった状態に割り付ける関数である。さて、地図データで記述されている走行路データから、上で述べたモデルを使って、ペトリネット構造Ｃ

Ｃ＝（Ｐ，Ｔ，Ｉ，Ｏ，ＧＩ，ＰＩ）

を生成するのが、図３１のペトリネット生成機構である。以下、ペトリネット生成機構による構造Ｃの作成手順を説明する。

❶プレース集合Ｐの作成

地図データから、ノードを順に取り出し、ノードとプレースを一対一対応させて、プレースｐ１，ｐ２，・・・，ｐｍを順に作る。

❷トランジション集合Ｔ，入出力アーク集合Ｉ，Ｏの作成

地図データに記述されてあるノード間の走行可能データから、始点ノードと終点ノードの組み合わせて全て取り出して、始点ノード，終点ノードに対応するプレースを、それぞれ入力プレース，出力プレースとするトランジションｔ１，ｔ２，・・・，ｔｎを順次作り出す。このとき同時に、入力アークの集合，出力アークの集合も作成される。

❸幾何抑止アーク集合ＧＩの作成

ノード同士が接近しすぎていて、それぞれのノードにロボットが同時に入ると衝突してしまう場合がある。このようなノードを干渉ノードという。干渉ノードは、ノードの座標とロボットの形状から幾何学的に算出できる。このような干渉ノードを算出して、ノードに対応するプレースを出力プレースとするトランジションから、干渉ノードに対応するプレースへ幾何抑止アークを付ける。

❹経路抑止アーク集合ＰＩの作成

走行路の形状によっては、ロボットの経路上に他のロボットがいるにも関わらず、次のノードに行くと、ロボット同士が邪魔し合ってなかなか動けなくなる場合がある。このような場合、次のノードに対応するプレースを出力プレースとするトランジションから、ここにロボットがいたら邪魔となるであろうノードに対応するプレースへ経路抑止アークを付ける。以下、地図を例に、構造Ｃがどのように作成されるかを説明する。

［例１］前述した図３の走行路において、ノード１，２，３，４からなるループは一方向、ノード３と５の間は両方向とする。この走行路では、干渉ノードは無いとする。上述の作成手順にしたがって、次のように構造Ｃが作成される。

❶プレース集合Ｐの作成

ノード１，２，３，４，５に対応させて、プレースｐ１，ｐ２，ｐ３，ｐ４，ｐ５を作って、

Ｐ＝｛ｐ１，ｐ２，ｐ３，ｐ４，ｐ５｝

とする。

❷トランジション集合Ｔ，入出力アーク集合Ｉ，Ｏの作成

始点ノードと終点ノードの組合せは、次のようになる。

｛(1,2)，(2,3)，(3,5)，(3,4)，(4,1)，(5,3)｝

したがって、これに対応する入力プレースと出力プレースの組合せは、

｛(ｐ1,ｐ2)，(ｐ2,ｐ3)，(ｐ3,ｐ5)，(ｐ3,ｐ4)，(ｐ4,ｐ1)，(ｐ5,ｐ3)｝

のようになる。この組合せ一つ一つにトランジションを対応させると、

Ｔ＝｛（ｔ12，ｔ23，ｔ35，ｔ34，ｔ41，ｔ53｝

のように求まる。以上から、入力アークの集合は

Ｉ＝｛(p1,t12)，(p2,t23)，(p3,t35)，(p3,t34)，(p4,t41)，(p5,t53)｝

となり、出力アークの集合は

Ｏ＝｛(t12,p2)，(t23,p3)，(t35,p5)，(t34,p4,)，(t41,p1)，(t53,p3)｝

となる。

③幾何抑止アーク集合ＧＩの作成

干渉ノードが無いので、幾何抑止アークも無い。

④経路抑止アーク集合ＰＩ作成

ノード５にロボットがいるときに、そこへ行こうとする他のロボットがノード２を経由して３まで来たとすると、３と５でロボット同士が向かい合うことになる。ノード５にいるロボットは動けないので、ノード３にいるロボットが一旦ノード４へ行って道を譲った後、再びノード１，２，３を通ってノード５へ行かなければならない。これは、非常に無駄である。このような場合は、ノード２で待たせればよい。これを実現するために、ｔ23からｐ5 へ、主経路抑止アークを付ける。以上を図示すれば、図３２のようになる。

［例２］前述した図５の走行路は、ノード数７で、線で結ばれた隣接ノード間は両方向である。そして、ノード２と３は接近し過ぎて干渉し合うとする。また、経路抑止アークはこの場合考えないとする。構造Ｃは次のように作成される。

①プレース集合Ｐの作成

ノード１，２，３，４，５，６，７に対応して、

Ｐ＝｛ｐ1，ｐ2，ｐ3，ｐ4，ｐ5，ｐ6，ｐ7｝

が作成される。

②トランジション集合Ｔ，入出力アーク集合Ｉ，Ｏの作成

始点ノードと終点ノードの組合せは、次のようになる。

｛(1,2)，(2,1)，(2,3)，(3,2)，(3,4)，(3,6)，(4,3)，(5,6)，(6,3)，(6,5)　(6,7)，(7,6)｝

したがって、これに対応する入力プレースと出力プレースの組合せは、

｛(p1,p2)，(p2,p1)，(p2,p3)，(p3,p2)，(p3,p4)，(p3,p6)，(p4,p3)，(p5,p6)，(p6,p3)，(p6,p5)，(p6,p7)，(p7,p6)｝

のようになる。この組合せ一つ一つにトランジションを対応させると、

Ｔ＝｛(t12,t21,t23,t32,t34,t36,t43,t56,t63,t65,t67,t76｝

のように求まる。以上から、入力アークの集合は、

Ｉ＝｛(p1,t12)，(p2,t21)，(p2,t23)，(p3,t32)(p3,t34)，(p3,t36)，(p4,t43)，(p5,t56)，(p6,t63)，(p6,t65)，(p6,t67)，(p7,t76)｝

出力アークの集合は

Ｏ＝｛t12,p2)，(t21,p1)，(t23,p3)，(t32,p2)，(t34,p4)，(t36,p6)，(t43,p3)，(t56,p6)，(t63,p3)，(t65,p5)，(t67,p7)，(t76,p6)｝

となる。

③幾何抑止アーク集合ＧＩの作成

干渉するノードは２と３である。したがって、ノード２に対応するプレースｐ2 を出力プレースとするトランジションを捜すと、ｔ12とｔ32がある。ここで、ｔ32の入力プレースはｐ3 で、これはノード２と干渉するノード３に相当する。したがって、ノード２から３へ向かう場合のトランジションなので、幾何抑止アークを付ける必要はない。よって、ｔ12からｐ3 へ幾何抑止アークが付く。同様に、ｐ3 を出力プレースとするトランジションｔ23，ｔ43，ｔ63のうち、ｔ23には幾何抑止アークを付ける必要がなく、ｔ43とｔ63からそれぞれｐ2 へ幾何抑止アークが付く。

④経路抑止アーク集合ＰＩの作成

今の場合、経路抑止アークは付かない。以上を図示すれば、図６のようになる。このように、ペトリネットモデルでは、走行方向も明示するので、走行路が分かりやすい。また、ノード同士の座標やロボット形状に依存する干渉ノードの関係が、幾何抑止アークで表現されるので、予約処理ロジックの妥当性を評価し易い。

【００１１】<第３実施例>本実施例は、移動ロボットシステムに対し、搬送路を記述した地図データを構造的に記述したペトリネット構造Ｃを図示し、その上に現在の予約状況やロボット現在位置をトークンによって表現するペトリネット表示機構を具備させたものである。

「どのロボットに、どこまでの経路を与えるか」を決める予約処理は、複数台のロボットをうまく走らせる基本ロジックである。しかし、この判断のもととなる「現在どのノードがどのロボットに予約されているか」（ノード予約状況）や「ロボットが現在どこにいるか」（ロボットの位置）といった状態を表わすデータは、制御局の内部メモリ上に保持されているだけで、構造的でなく、実行中に見れない。このため、予約処理の妥当性が把握しにくい。そこで、本実施例においては、現在のノード予約状況やロボットの位置を、図示するようにした。図３３は本実施例の機能を示すブロック図である。上記第２実施例において、ペトリネット生成機構により、

Ｃ＝（Ｐ，Ｔ，Ｉ，Ｏ，ＧＩ，ＰＩ）

のペトリネット構造を作った。ただ、それのみではメモリ上のデータとして保持されるだけなので、まだ分かりにくい。そこで、上記第２実施例のように、ペトリネット図で表現すれば理解し易い。本実施例は、構造Ｃをこのように図示することを基礎にする。ただし、Ｃだけでは、現在のノード予約情況やロボットの位置といった状態が記述できないので、Ｃの上にマーキングＭを加えることにより状態も併せて表現する。したがって、ペトリネットＰＮ

ＰＮ＝（Ｃ，Ｍ）

で構成される。制御局は運用中にロボットと通信しているので、ロボットが現在どこにいるかは把握している。また、ノードの予約処理は制御局自身が行なうので、どのノードが予約されているかも知っている。これから、ノードｎの現在の状態に応じて、ノードｎに対応するプレースｐnのマーキング値が以下のように定まる。

ノードｎにロボットがいる　　Ｍ（ｐｎ）＝２

ノードｎが予約されている　　Ｍ（ｐｎ）＝１

したがって、M（pn）の値に応じて、白トークン（M（pn）＝1），黒トークン（M（pn）＝2）をプレースpn の中に表示すれば、ノード予約状況やロボットの位置を、図示することができる。構成を、図33に示す。ペトリネット表示機構は、上記第2実施例によるペトリネット構造Cと、制御局が保持しているマーキング状態Mを参照して、ペトリネット図を画面上に表示する。以下、本実施例の具体的動作を上記第2実施例の例1を使って説明する。

①最初に、ロボット1と2がノード1と5で待機しているとする。どちらも止まっているので、予約ノードはない。したがって、p1 とp5 に黒トークンがある。（図34）

②しばらくして、ロボット1がノード5を目標としノード2を予約して、ノード1から出発したとする。この場合、p1 とp5 に黒トークンp2に白トークンが表示される。（図35）

③次に、ロボット1がノード2に到着したとする。p1の黒トークンとp2 の白トークンが消え、p2 に黒トークンが現われる。（図36）

④ロボット2は、ロボット1を邪魔としているので、邪魔にならないノード4を目指して走りはじめる。この場合、ノード3と4を予約するとすると、図37のようになる。

⑤ロボット2がノード3に到達すると、図38のようになる。

⑥ロボット2がノード4に到達すると、図39のようになる。

⑦邪魔なロボットがいなくなったので、ロボット1はノード3と5を予約して走り出す（図40）。

⑧ロボット1がノード3に到達すると、図41のようになる。

⑨ロボット1がノード5に到達すると、図42のようになる。

以上説明したように、予約処理の構造を記述したペトリネット構造Cの上に、現在の予約状況やロボット現在位置を表示するので、状態を把握し易く、予約処理ロジックの妥当性も評価し易い。

【0012】<第4実施例>本実施例の構成を、図43に示す。第2実施例の手法を使って、走行路の状態も含めた予約処理のロジックを、ペトリネット構造として表現しておく。さらに、第3実施例の手法を使えば、予約処理ロジックと走行路状態が図示できるので、さらに理解し易い。ロボットからの経路要求がくると、第2実施例で述べたようにして、トランジションの発火系列に変換する。この系列は、既に予約済みのノード列に相当する既発火系列と、これから予約を取りたいノード列に相当する未発火系列に分けられる。ペトリネット発火機構は、この未発火系列のトランジションを順次取り出し、発火可能なら発火させる。トランジションが発火すると、マーキング状態が変化する。トランジションの発火可能条件と発火規則は、上記第1実施例において説明したものを用いる。第3実施例の場合と同じ場合を想定し説明する。

（1）第3実施例において①から②へ移る際に、ロボット1はノード2と3を制御局に要求するとする。この場合、未発火系列は｛t12，t23｝である。最初にt12について、第1実施例で説明した発火可能条件を調べる。この場合、発火可能条件①から④を全て満足するので、t12は発火可能である。そこでt12が発火すると、t12の出力プレースp2 に白トークンが現われる。

（2）続いて、t23が発火可能か調べるが、t23の主経路抑止アーク先のp5 はロボット1の経路上でかつ、p5 に他のロボット2がいる（当然、予約されている）ので、発火可能条件④は満たされない。したがって、t23は発火不可能である。

（3）第3実施例の④で走り始める前に、ロボット2はノード3と4を制御局に要求する。この場合、未発火系列は｛t53，t34｝である。t53は発火可能なので、発火する。そうすると、p3 に白トークンが現われるので、続いてt34も発火できp4 にも白トークンが現われる。

（4）第3実施例の⑦でも走り始める前に、ロボット1はノード3と5を制御局に要求する。この場合、未発火系列は｛t23，t35｝である。（3）のケースと同様で、t23，t35が順次発火する。

このように、ノードの予約処理条件を構造的に表現したペトリネットモデル上で、トランジションの遷移則（発火可能条件と発火処理）によって一般的かつ規則的に予約処理が行えるので、予約処理の妥当性も評価し易い。

【0013】

【発明の効果】以上説明したように、この発明によれば、衝突あるいは衝突が起り易い状態になったのを検知し、その原因となった各移動ロボットの配置状態および進行状態を検出する原因検出手段と、前記原因検出手段によって検出された原因を抑止情報として記憶する抑止情報記憶手段と、前記抑止情報に基づいて前記移動ロボットの移動を制限し、前記移動ロボットの移動制御を行う制御手段とを設けたので、衝突あるいは衝突が起り易い状態になることが早期に防止される。従って、移動ロボットシステムの効率が向上するという効果が得られる。

【図面の簡単な説明】

【図1】この発明の第1実施例による移動ロボットシステムの構成を示すブロック図である。

【図2】　走行路を例示する図である。

【図3】　走行路を例示する図である。

【図4】　同走行路のペトリネットモデルを示す図である。

【図5】　走行路を例示する図である。

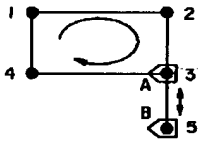
19

【図6】　同走行路のペトリネットモデルを示す図である。
【図7】　同実施例における経路抑止条件を説明する図である。
【図8】　同実施例におけるシミュレータを示すブロック図である。
【図9】　同シミュレータによるシミュレーションを示す図である。
【図10】　同シミュレータによるシミュレーションを示す図である。
【図11】　同シミュレータによるシミュレーションを示す図である。
【図12】　同シミュレータによるシミュレーションを示す図である。
【図13】　同シミュレータによるシミュレーションを示す図である。
【図14】　同シミュレータによるシミュレーションを示す図である。
【図15】　同シミュレータによるシミュレーションを示す図である。
【図16】　同シミュレータによるシミュレーションを示す図である。
【図17】　同シミュレータによるシミュレーションを示す図である。
【図18】　同シミュレータによるシミュレーションを示す図である。
【図19】　同シミュレータによるシミュレーションを示す図である。
【図20】　同シミュレータによるシミュレーションを示す図である。
【図21】　同シミュレータによるシミュレーションを示す図である。
【図22】　同シミュレータによるシミュレーションを示す図である。
【図23】　同シミュレータによるシミュレーションを示す図である。

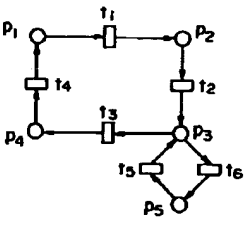
20

【図24】　同シミュレータによるシミュレーションを示す図である。
【図25】　同シミュレータによるシミュレーションを示す図である。
【図26】　同シミュレータによるシミュレーションを示す図である。
【図27】　同シミュレータによるシミュレーションを示す図である。
【図28】　同シミュレータによるシミュレーションを示す図である。
【図29】　同シミュレータによるシミュレーションを示す図である。
【図30】　同シミュレータによるシミュレーションを示す図である。
【図31】　この発明の第2実施例による移動ロボットシステムの機能を示すブロック図である。
【図32】　同実施例におけるペトリネットモデルを例示する図である。
【図33】　この発明の第3実施例による移動ロボットシステムの機能を示すブロック図である。
【図34】　同実施例の動作を説明する図である。
【図35】　同実施例の動作を説明する図である。
【図36】　同実施例の動作を説明する図である。
【図37】　同実施例の動作を説明する図である。
【図38】　同実施例の動作を説明する図である。
【図39】　同実施例の動作を説明する図である。
【図40】　同実施例の動作を説明する図である。
【図41】　同実施例の動作を説明する図である。
【図42】　同実施例の動作を説明する図である。
【図43】　この発明の第4実施例による移動ロボットシステムの機能を示すブロック図である。
【符号の説明】
C　　制御局
R　　移動ロボット
L　　走行路

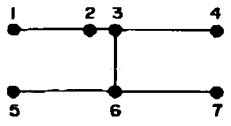
【図3】



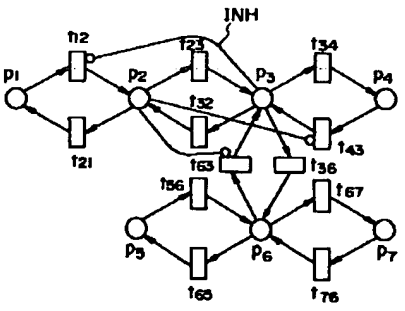
【図4】



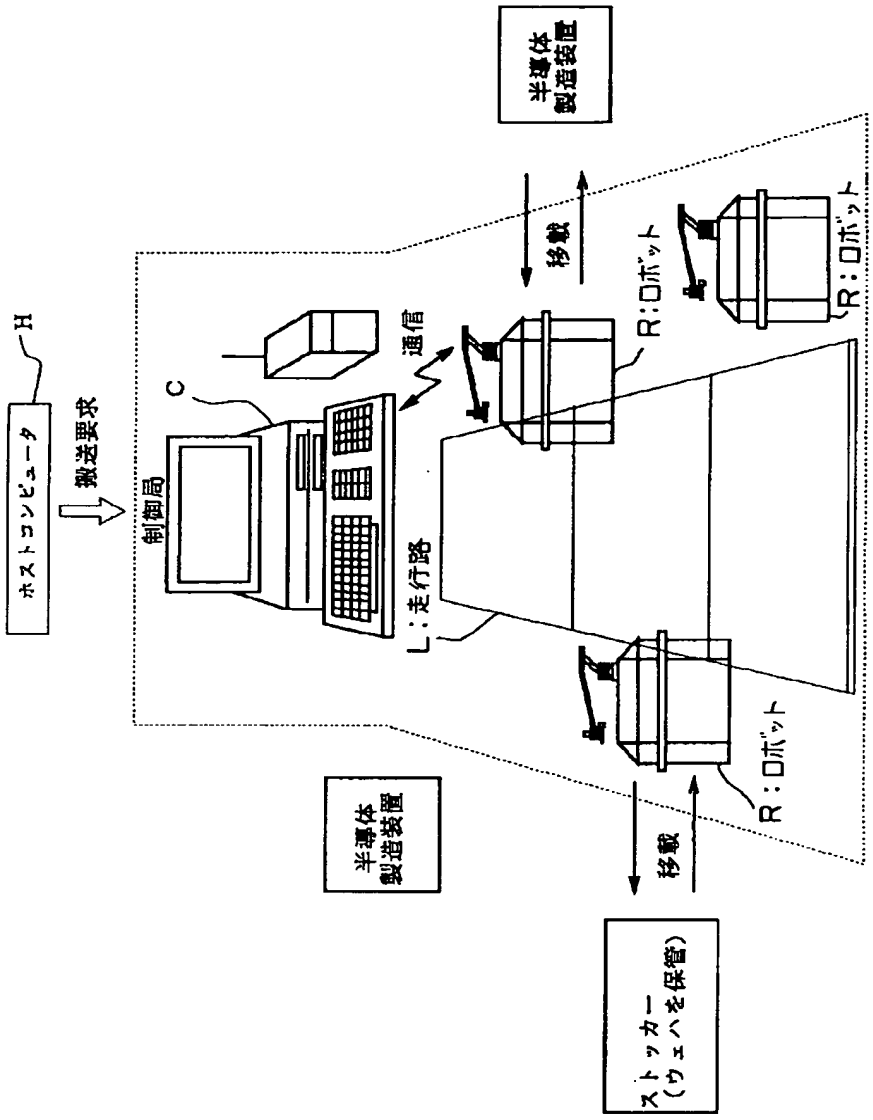
【図5】



【図6】

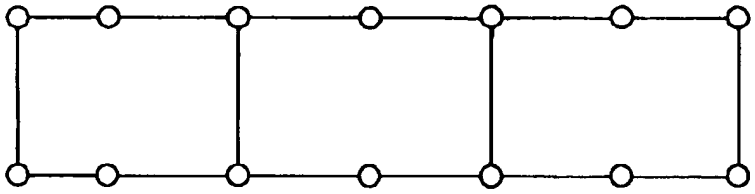


【図1】

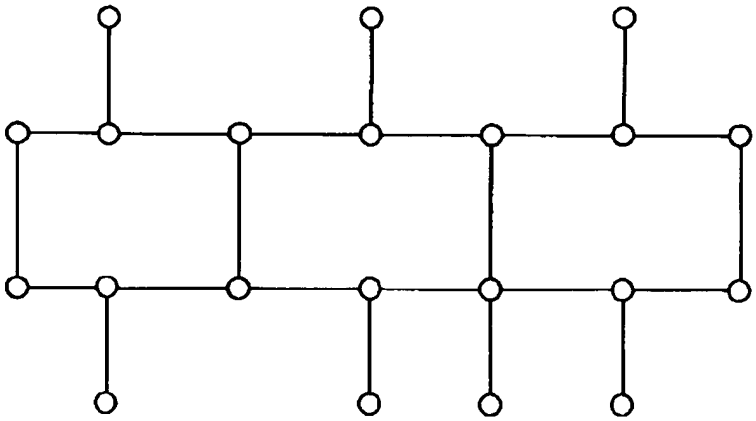

半導体
製造装置
移載
R:ロボット
R:ロボット
通信
H
ホストコンピュータ
搬送要求
C
制御局
L:走行路
R:ロボット
半導体
製造装置
移載
ストッカー
(ウェハを保管)

【図2】

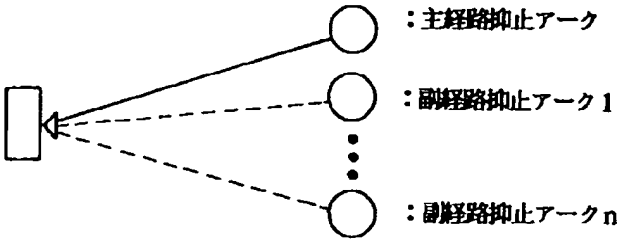
（a）梯子型走行路



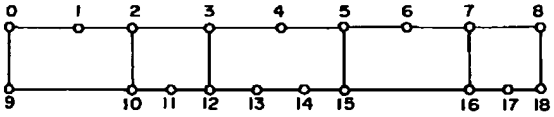
（b）ループ型走行路



【図7】



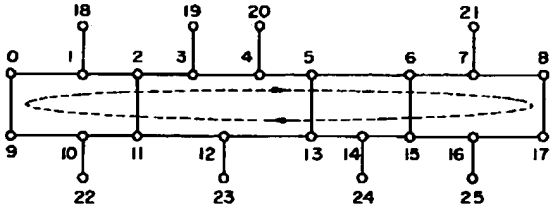
【図9】



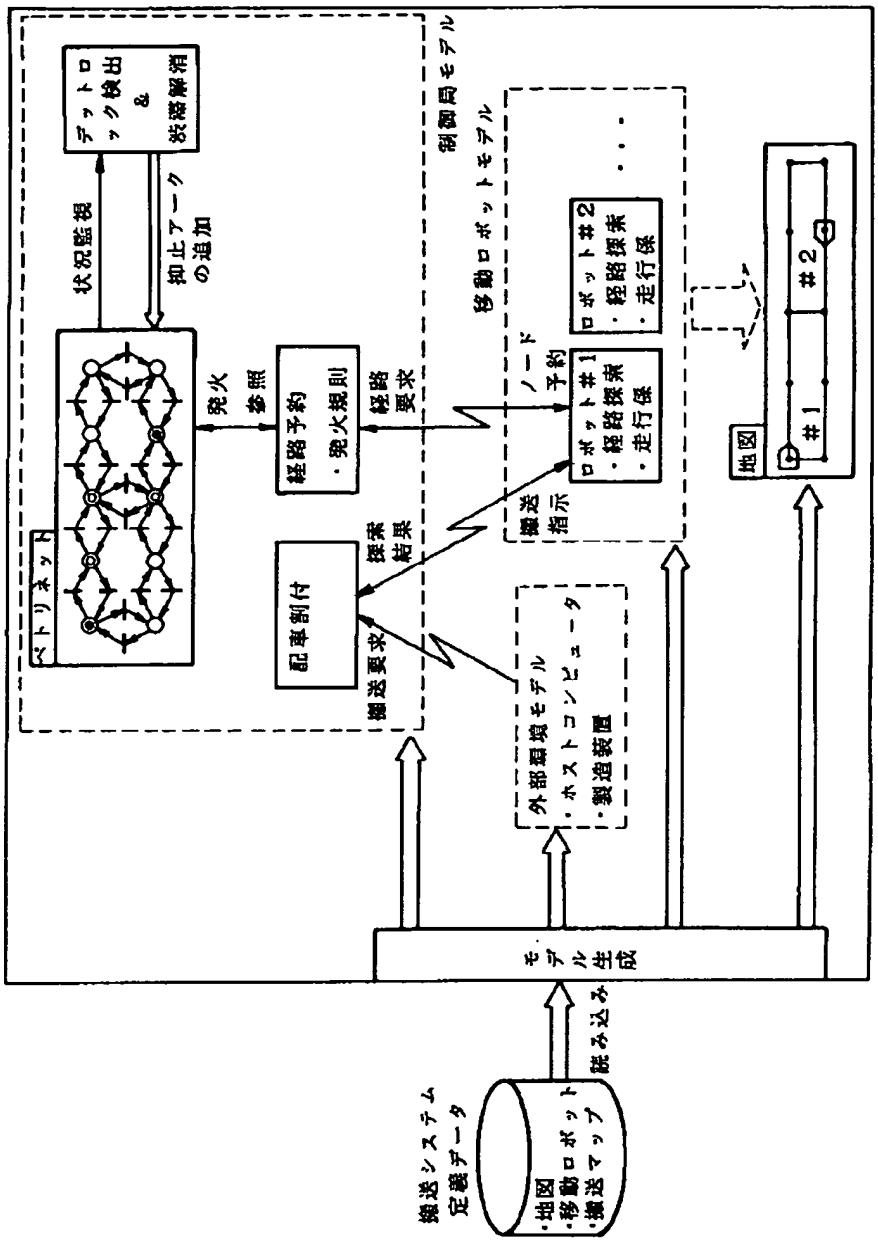
【図10】

fromノード	toノード	搬送回数
1	17	16
11	6	18
8	11	8
13	1	8
6	14	8
17	11	8
8	1	8

【図11】



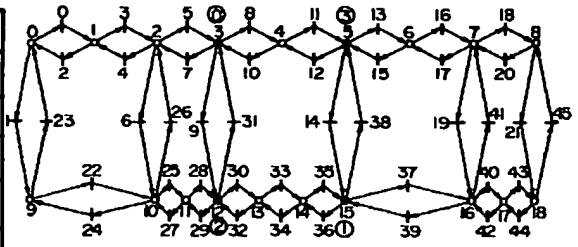
【図8】


制御局モデル
移動ロボットモデル
デットロック検出 & 救済解消
状況監視
抑止アークの追加
ペトリネット
発火参照
経路予約・発火規則
経路要求
ノード予約
ロボット#1・経路探索・走行係
ロボット#2・経路探索・走行係
搬送指示
探索結果
配車割付
搬送要求
外部環境モデル・ホストコンピュータ・製造装置
地図
#1
#2
モデル生成
読み込み
搬送システム定義データ
・地図
・移動ロボット
・搬送マップ

【図12】

fromノード	toノード	搬送回数
8	24	16
23	21	8
22	20	16
19	25	16
21	18	8
19	23	8
22	21	8

【図13】

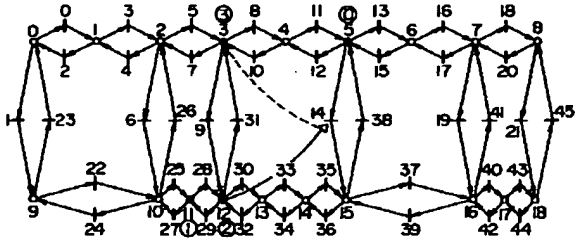


【図14】

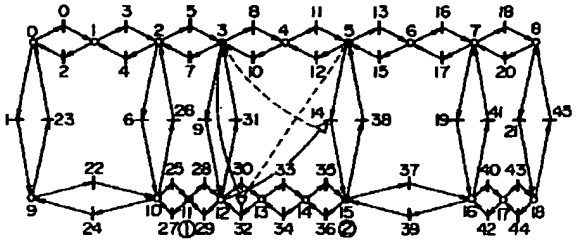
ロボット	Place	Obstacle	Old-Path	LFT
0	3	3	4→5→15→16→17	5
1	15	2	14→13→12→11	39
2	12	0	3→4→5→6→7→16→17	28
3	5	1	15→14→13	15

（ロボット1は作業中）

【図15】



【図17】



【図16】

ロボット	Place	Obstacle	Old-Path	LFT
0	5	2	15→14→13→12→11	15
1	11			
2	12	3	3→2→1	32
3	3	0	4→5→6	5

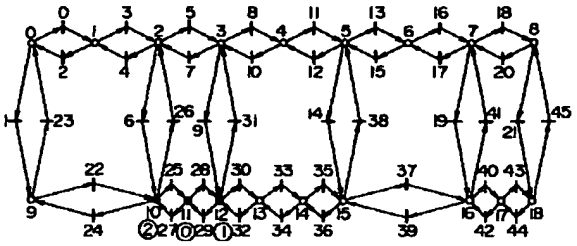
（ロボット1は作業中）

【図18】

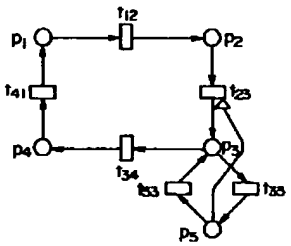
ロボット	Place	Obstacle	Old-Path	LFT
0	5	2	15←14←13←12←11	15
1	11			
2	15	3	14←13←12←3←2←1	39
3	3	0	4←5←6	5

（ロボット1は作業中）

【図19】



【図32】

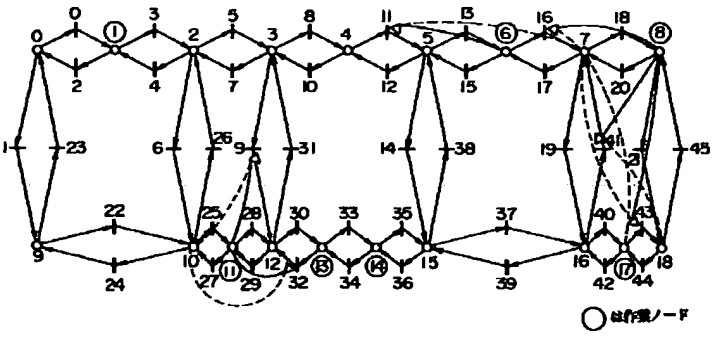


【図20】

ロボット	Place	Obstacle	Old-Path	LFT
0	11	1	12←3←4←5←6	
1	12	0	11	32
2	10	0	11	6

（ロボット1は作業中）

【図21】



【図28】

経路抑止アーク表

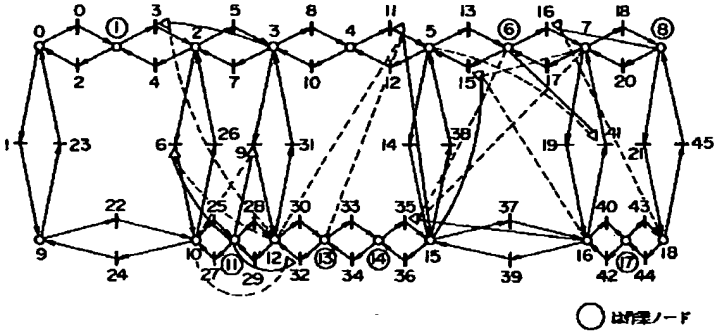
トランジション番号	主	副
0	18	
3	19	
5	20	
11	21	
19	22	
23	23	
27	24	

【図22】

経路抑止アーク表

トランジション番号	主	副
9	11	10
11	6	7
16	8	17
32	11	10
41	8	18
43	8	7

【図23】



【図24】

経路抑止アーク表

トランジション番号	主	副
3	3	12
6	11	12
9	11	10
11	15	12，13
15	15	7，16
16	6	16
32	11	10
35	16	6，7
41	6	5

【図26】

経路抑止アーク表

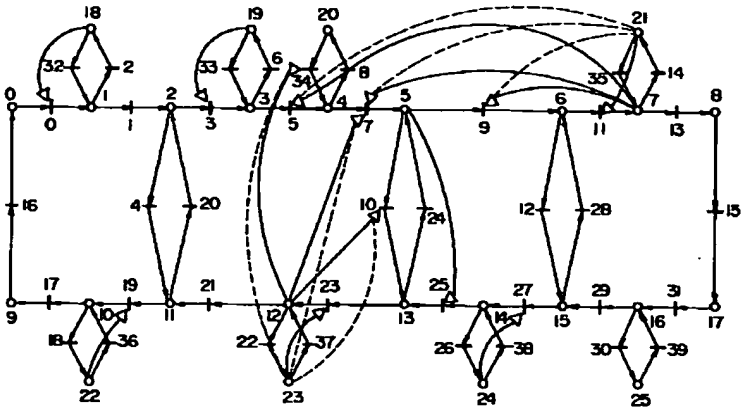
トランジション番号	主	副
0	18	
3	19	
5	7	21
7	7	21
7	12	23
9	7	21
10	12	23
11	21	
19	22	
23	23	
25	5	
27	24	
34	12	23

【図30】

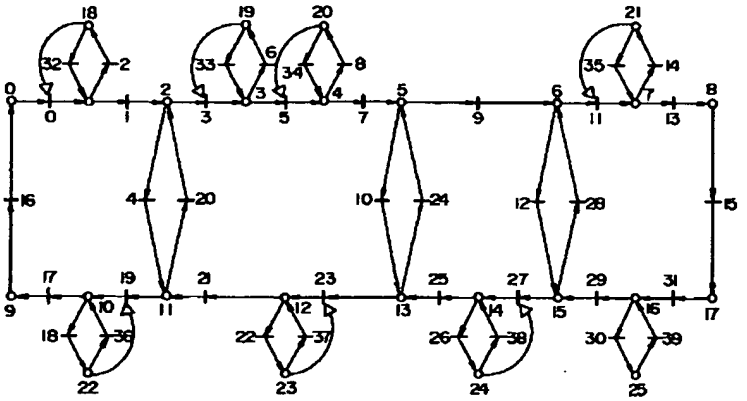
経路抑止アーク表

トランジション番号	主	副
0	18	
3	19	
5	20	
11	21	
19	22	
21	10	22
23	23	
25	12	23
27	24	
31	25	
37	10	22

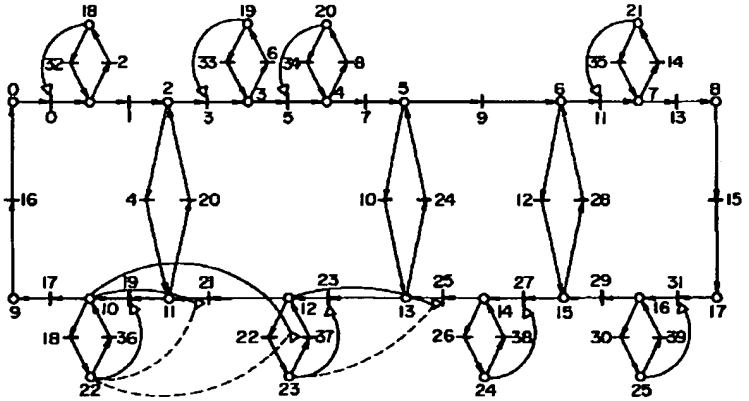
【図25】



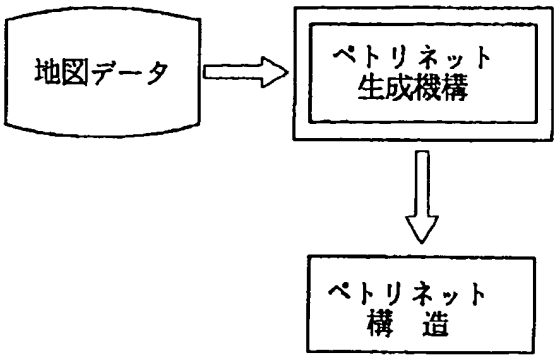
【図27】



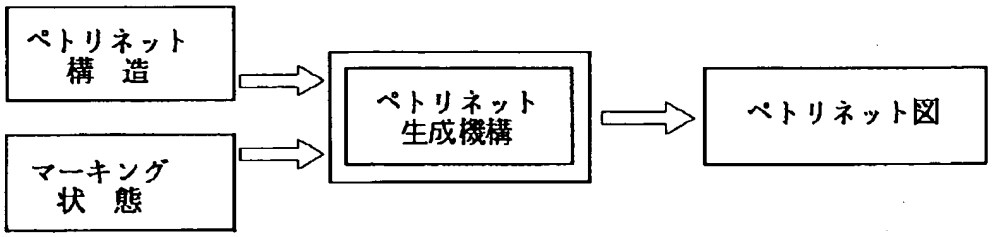
【図29】



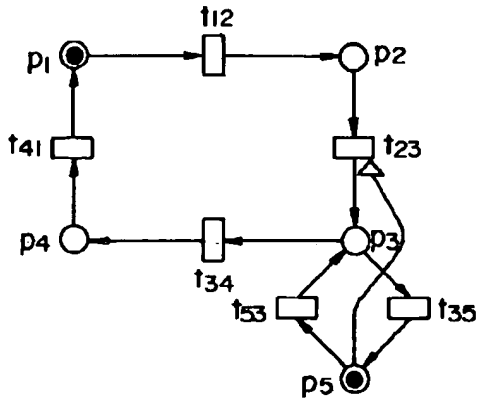
【図31】



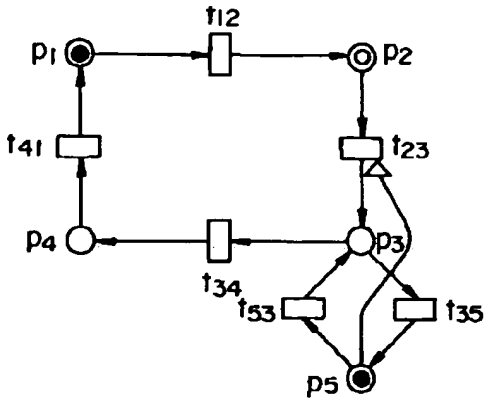
【図33】



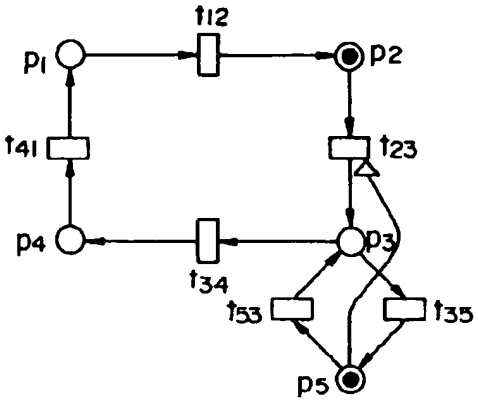
【図34】



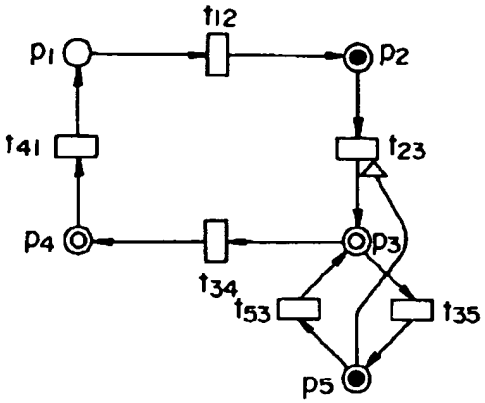
【図35】



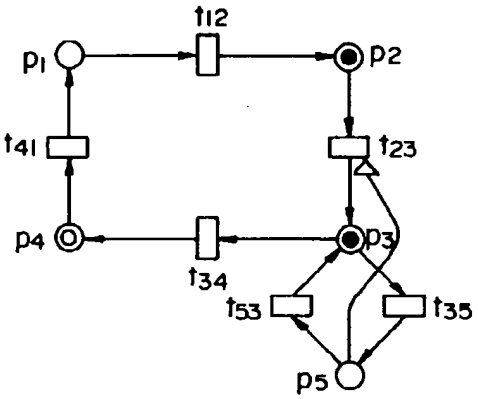
【図36】



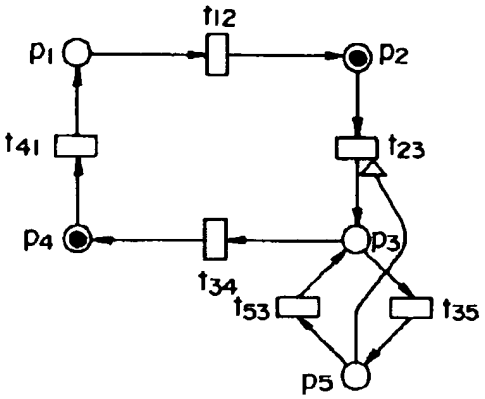
【図37】



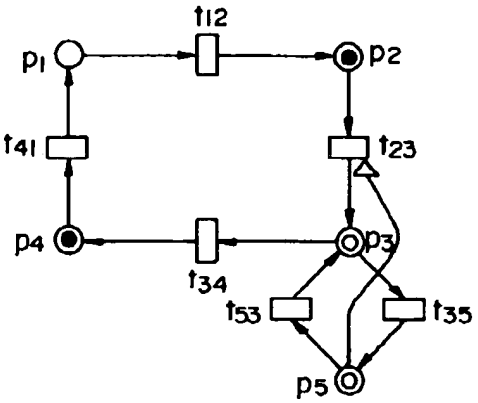
【図38】



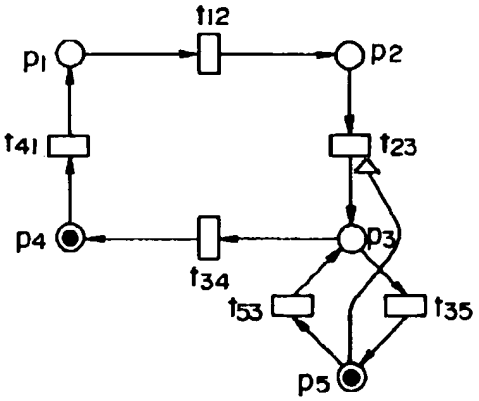
【図39】



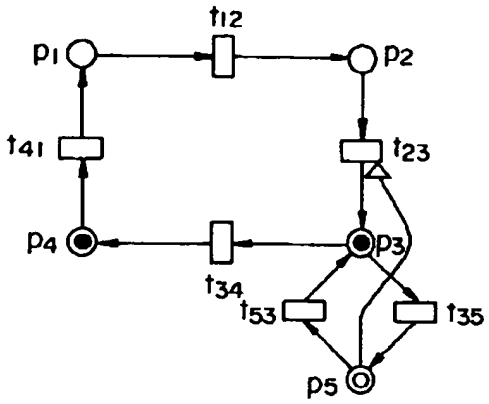
【図40】



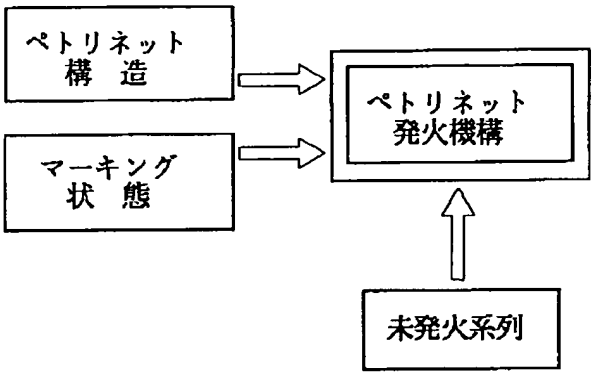
【図42】



【図41】


t12
p1
p2
t41
t23
p4
p3
t34
t53
t35
p5

【図43】


ペトリネット
構　造
マーキング
状　態
ペトリネット
発火機構
未発火系列

PAT-NO: JP405073143A

DOCUMENT-IDENTIFIER: JP 05073143 A

TITLE: MOBILE ROBOT SYSTEM

PUBN-DATE: March 26, 1993

INVENTOR-INFORMATION:
NAME
EGAWA, TAKAMI
TAKAMATSU, CHIAKI

ASSIGNEE-INFORMATION:

NAME	COUNTRY
SHINKO ELECTRIC CO LTD	N/A

APPL-NO: JP03121495

APPL-DATE: May 27, 1991

INT-CL (IPC): G05D001/02, G05B013/02

US-CL-CURRENT: 901/1

ABSTRACT:

PURPOSE: To prevent congestion or deadlock by early preventing collision from being generated by limiting the move of a robot based on stored suppression information and controlling the move of the mobile robot.

CONSTITUTION: A carrying request (job) is inputted from a host computer H or the like to a control station C. The control station C selects any proper job out of the jobs incompleting car distribution up to the moment, selects any

optimum robot out of robots R on standby and allocates the job. Each robot R
stores route data describing a traveling route L in a built-in memory. The
robot R allocated to the car distribution decides the route to the destination
based on the map data specifying the traveling route L and travels not to
collide the other robot according to the decided route. In this case, when a
state is turned to the collision or to easily generate the collision, the cause
is detected and stored as the suppression information. Afterwards, the mobile
robot is controlled not to collide or not to turn to the state of easily
generating the collision by the same cause.